Zusammenfassung der Experimente mit dem alten Raytracer

Um sich einen ersten Eindruck von der Komplexitaet und dem Data-Flow des Raytracing Algorithmus zu verschaffen wurden einige Experimente mit dem alten Raytracer gemacht.
Bei den Untersuchungen stellte sich heraus das ein wichtiger Teil des Raytracing Algorithmus das Tracing, die Verfolgung der Strahlen bis zu den Objekten, und die Intersection Tests auf zwei verschiedenen SPUs ausgefuehrt wurde.
Das fuehrte dazu das Objekte die in zwei Voxeln residieren zweimal mit dem selben Strahl geschnitten wurden. Denn das TRAV SPUlet wusste es gibt Objekte im aktuellen Voxel, aber nicht wo der Schnittpunkt mit dem Objekt ist, da erst das naechste, das INT SPUlet die Intersection Tests ausfuehrt.

Untersuchungen zeigten das ein nicht unerheblicher Teil der traversierten Strahlen nach den Intersection Tests an die TRAV SPUlet zurueckgereicht wurden um erneut traversiert zu werden, weil der Schnittpunkt erst im naechsten bzw. uebernaechsten Voxel ist. (Im schlimmsten Fall erstreckt sich das Objekt fast parallel zum Strahl und wird erst viele Voxel spaeter vom Strahl getroffen).

Folgende Tabelle zeigt auf wieviel Strahlen zum Traversieren zurueckgereicht werden um erneut traversiert und mit den Objekten geschnitten zu werden.

Dim. Bild	Rays Gesamt	Rays die ein Objekt treffen	Rays die zurueckgereicht werden	Prozensatz
64x64	4096	1556	1011	~64.7%
256x256	65536	34984	17923	~51 %
2400x2400	5760000	2685992	2322923	~86,4%

Sieht man sich die Zahlen an erkennt man sofort das ein grosser Prozentsatz der Strahlen zurueckgereicht wird. Zu den doppelten Intersection Tests kommt noch der Kommunikations-Overhead hinzu um die Strahlen zu Lesen und zu Schreiben.

Nach zusammenfuehren der beiden SPUlets konnte man ein Zeitgewinn von 32 % verzeichnen.

Test: ./1200 0 0 10 8 Original: 77.36 sec Neuer Wert: 52.72 sec

(S_TRAV und S_INT wurden auch zusammengefuehrt 2-3 sec Zeitgewinn)
Whitted: 75 - 95 % der Raytracing Zeit sind Intersection tests!

Um die Intersection Test zu beschleunigen wurden die Algorithmen vektorisiert. Der Vorteil von vektorisierten Code ist das mit einer Instruktion 4 (float) Variablen gleichzeitig berechnet werden koennen. Geometrische Intersection Tests arbeiten mit Vektoren die praedestiniert sind fuer Vektorcode. Anstatt jede einzelne Koordinate einzeln zu berechnen kann das jetzt mit einer einzigen Instruktion geschehen. Desweiteren faellt der Overhead weg die Variablen richtig in die Register zu platzieren falls sie nicht aligned sind.

Test haben gezeigt das sich die Anzahl der Instruktionen auf die Haelfte reduzieren laesst und der Code um einiges schneller laeuft.

Test: ./1200 0 0 10 8 Original: 77.36 sec Neuer Wert: 43.68 sec

Ein weiterer Knackpunkt beim alten Raytracer ist die Kommunikation mit dem Arbeitsspeicher. An manchen Stellen werden nur einzelne Objekte aus dem AB geholt obwohl gleich die ganze List auf einmal in den Local Store geladen werden koennte. Das fetchen der ganzen Liste verringert den Overhead der Kommunikation und verringert die Anzahl der stall cycles.

Weitere Untersuchungen zeigten das mit gezielten branch hints der Raytracer um 1 bis 2 sec schneller laeuft.

Am Ende der Experimentierphase erreichte der neue Raytracer eine Zeit die um den Faktor 2 kleiner war.

Test: ./1200 0 0 10 8 Original: 77.36 sec Neuer Wert: 35.42 sec